Exploiting Value Locality through Physical Register Reuse and Unification

-or-

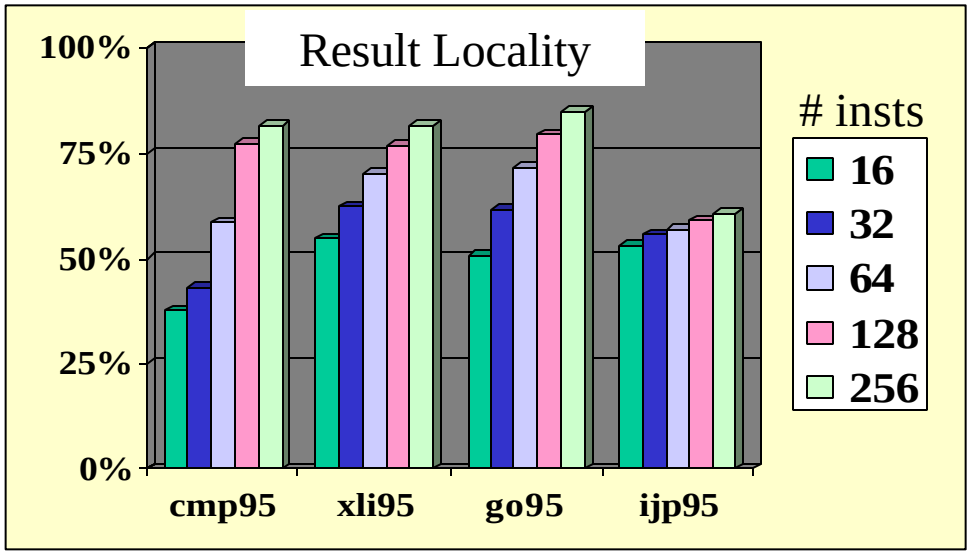
One Value, One Location

Stephan Jourdan, Ronny Ronen, Michael Bekerman, Bishara Shomar, and Adi Yoaz

Intel Corporation

Triggers

High Redundancy in Result Values



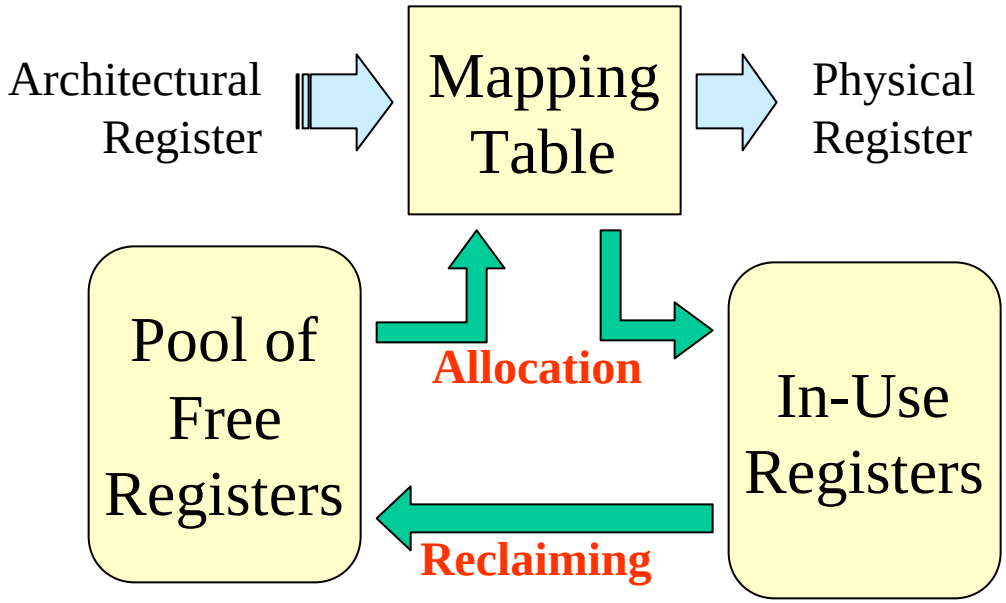
Let's Reuse Storage Locations

Outline

- **Motivation**
- **Current Schemes**
  - Register
  - Memory
- **Physical Register Reuse**
  - Support
  - Benefits
  - Value Matching
  - Value Identity Detection
- **Unification**
  - Memory Dependencies
  - Verification
  - Load-to-Load
- **Conclusion**
  - x86 ISA

Current Scheme: REGISTER Dependency Tracking and Renaming

- **Tracking:** **Retrieve last producers**
- **Renaming:** **Provide multiple locations**

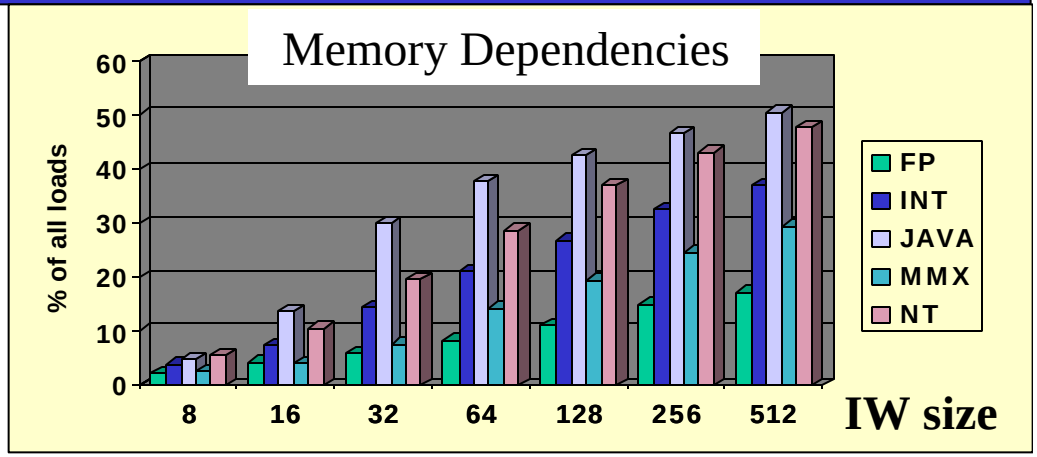


Physical registers (PRs):

- Decoupling
- Provide storage
- Mark dependencies for scheduling

Current Scheme: MEMORY Dependency Tracking and Renaming

- **Memory Dependencies (IW)**
  - Ordering Buffer (T)
  - Forwarding Buffer (R)



- **Problem: Dynamic definition of the locations**

  Solution: *Introduce fake register dependencies:*
  - Loads w/ respect to all older Stores
  - None
  - Predicted

Physical Register Reuse (PRR)

Idea

Reuse a physical location whenever it is detected that an incoming result value matches a previous one.

Performed before renaming!

Supporting PRR

- **Problem: Reclaiming**
- **Add counters to all PRs**
  - incremented on each allocation
  - decremented on each reclaiming
  - if (ctr==0) send to Free List

PRR Benefits

- **Sharing (Storage)**

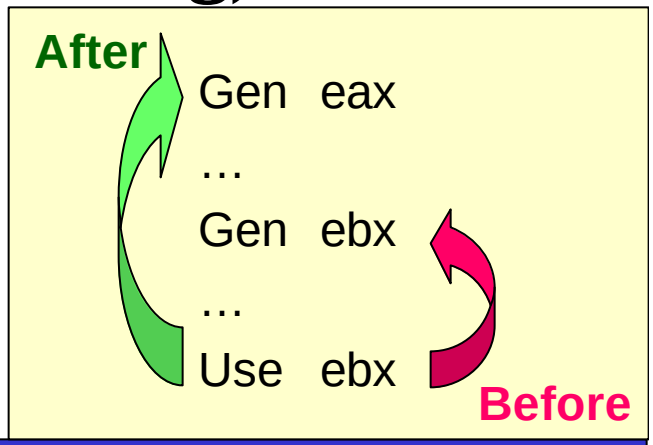
  Requires less PRs

  - Increase IW size
  - Decrease Register File access time

- **Dependency Redirection (Scheduling)**

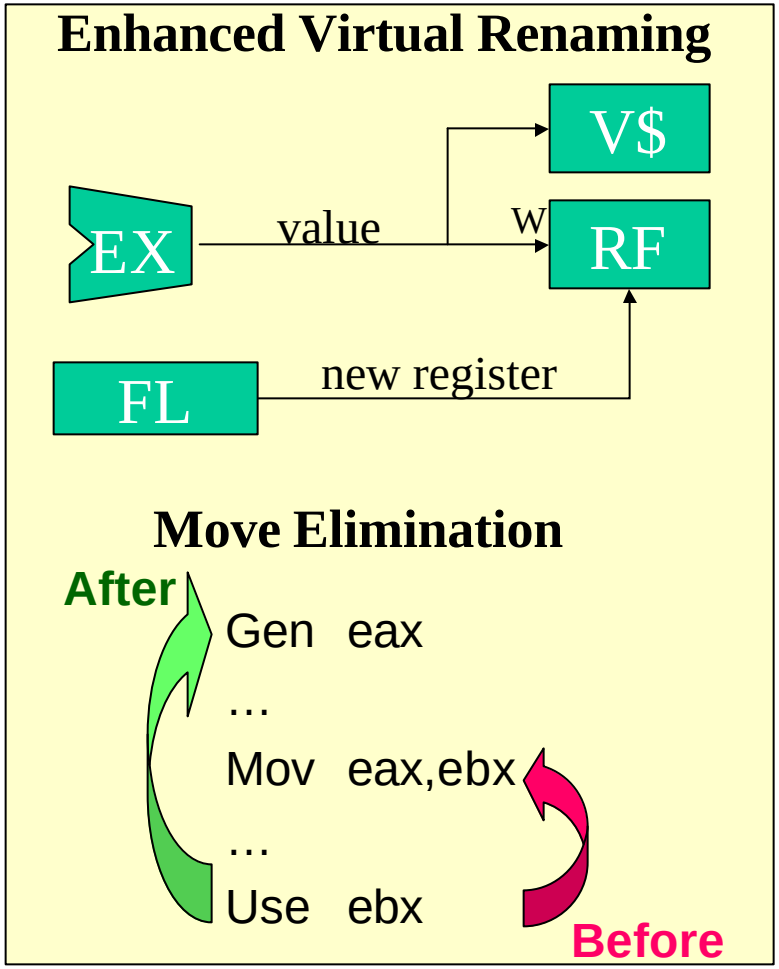
  Dynamically Changes DFG (iDFG)

  - Boost ILP



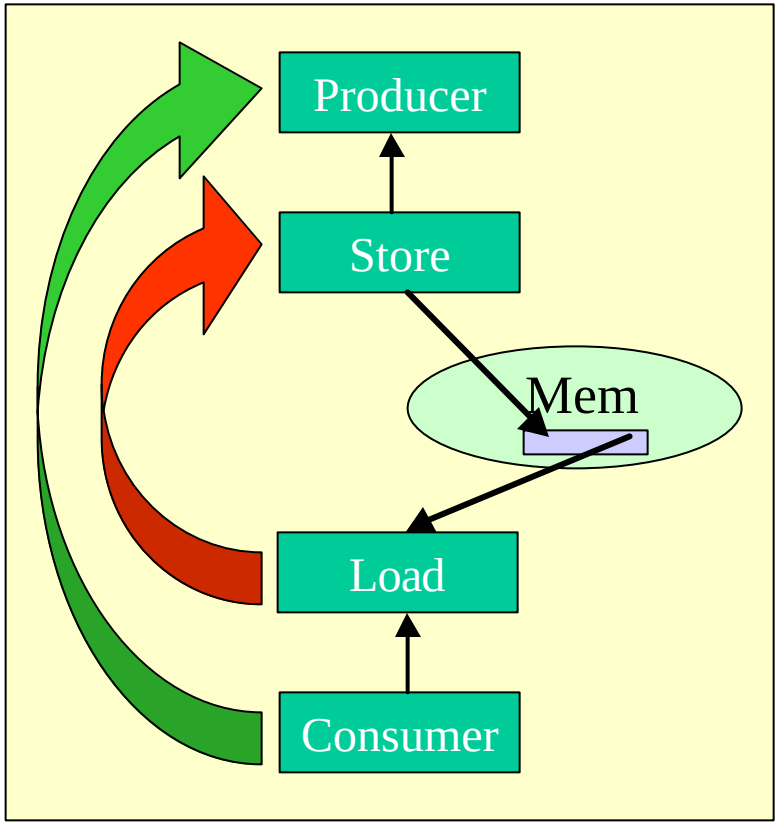
Performing PRR

- **Value Matching**
  - Look-up into a Value Cache
    - Prediction Table
    - Reuse Buffer
    - Decoder
- **Value-Identity Detectors**
  - Either speculative or safe
    - Memory Dependencies
    - Load-load



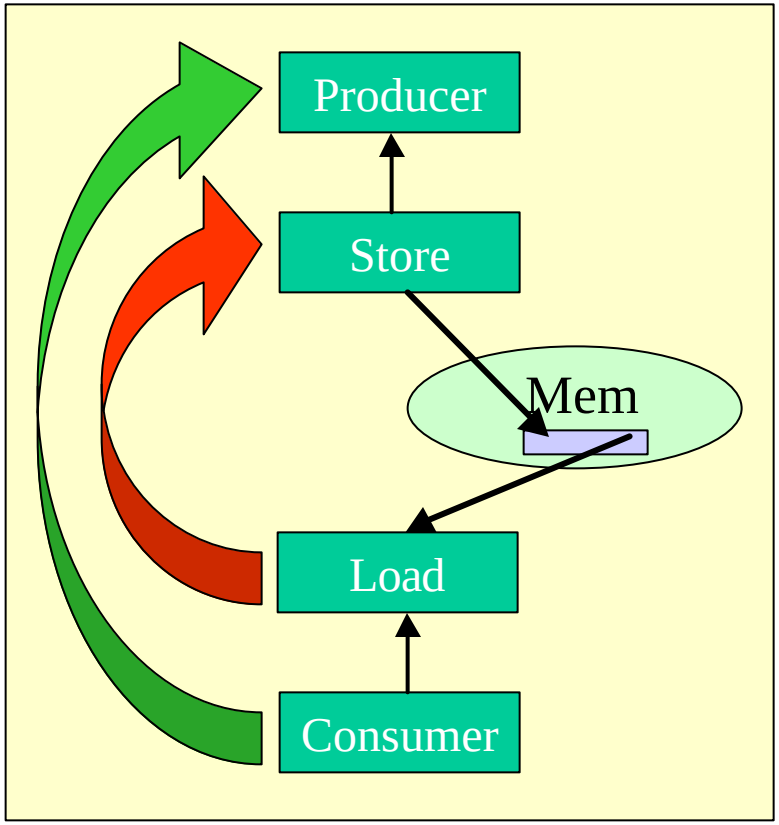
PRR: Unified Renaming

- **Register Dependencies**
- **Memory Dependencies**
  - Dynamic Register Dependencies
- **Unified Renaming**

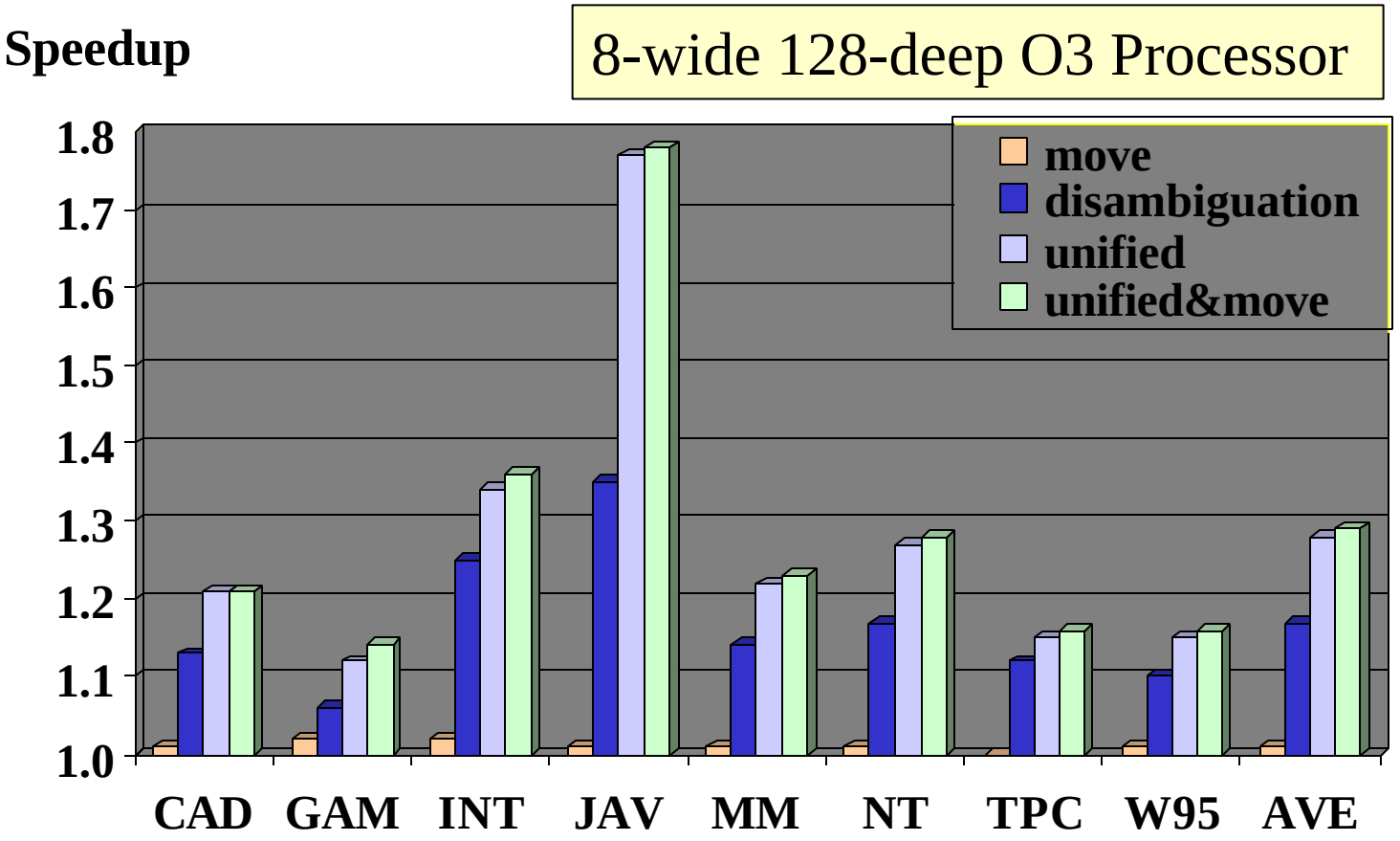


Unified Renaming: Verification

- **Value-Identity Prediction**
  - Load Res. == Producer Res.
  - checking done w/ values
- **Early Checking: Addresses**
  - Load Add. == Store Add.
  - cut 1/3 of the load traffic
- **Load-to-Load (2/3 traffic)**



Performance Improvement



Conclusion

- **Physical Register Reuse**
  - One Value, One Location
- **Dependency Redirection**
  - Dynamic DFG Translation
- **Unified Renaming**
  - No Memory Dependencies